Attorney Docket No. 42390.P11491 PATENT

UNITED STATES PATENT APPLICATION

FOR

METHOD AND APPARATUS FOR MODIFYING THE CONTENTS OF REVISION IDENTIFICATION REGISTER

Applicants:

Rajeev K. Nalawadi
Faraz A. Siddiqi

Prepared by:

Calvin E. Wells
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
2200 Mission College Boulevard
Santa Clara, CA 95052

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number EL867648893US
Date of Deposit December 31, 2001
I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Michelle Begay
(Typed or printed name of person mailing paper or fee)

Michelle Begay
(Signature of person mailing paper or fee)

METHOD AND APPARATUS FOR MODIFYING THE CONTENTS OF A REVISION IDENTIFICATION REGISTER

Field Of The Invention

[0001] The present invention pertains to the field of computer systems. More particularly, this invention pertains to the field of revision identification registers.

Background of the Invention

[0002] In a typical computer system, many of the computer system devices include revision identification registers that contain a value that indicates the current revision of the associated hardware. Typically, every time a device undergoes a revision (often referred to as a "stepping"), the value stored in the revision identification register is changed to reflect the new stepping. Prior computer system devices implement the revision identification registers as read-only registers. In other words, the contents of the revision identification registers are not modifiable.

[0003] When an operating system is loaded during a computer system boot process, the operating system typically checks the various revision identification registers to determine the current steppings of the various devices. The operating system uses the revision identification information to make decisions regarding which device drivers to load. When the operating system recognizes a new device stepping, the operating system loads an updated device driver for that device.

[0004] Computer systems typically include highly-integrated system logic devices often referred to as "chipsets". Theses system logic devices may include many functional units, and many of these functional units may include revision identification registers. Typically, when a highly-integrated system logic device undergoes a new stepping, most of the functional units do not have a change in functionality to warrant a change in the device driver. However, for each new

stepping, each of the revision identification registers associated with the integrated functional units contain a value that reflects the new stepping. During the boot process, if the operating system detects a new device stepping, the operating system will load updated device drivers for the various functional units even if there is no change in functionality from the previous stepping to the current stepping.

[0005] The loading of updated drivers whether needed or not when a new stepping is detected poses difficulties for computer system manufacturers. Computer system manufacturers often build a pre-load of an operating system on one system and then moves it to an identical system that may have a system logic device with a different silicon stepping. Upon power-up, the operating system will load new device drivers even if not needed, thereby increasing manufacturing time.

[0006] The revision identification registers described above are often implemented in accordance with the Peripheral Component Interconnect (PCI) bus standard (Peripheral Component Interconnect Local Bus Specification, Rev. 2.2, released December 18, 1998).

Brief Description of the Drawings

[0007] The invention will be understood more fully from the detailed description given below and from the accompanying drawings of embodiments of the invention which, however, should not be taken to limit the invention to the specific embodiments described, but are for explanation and understanding only.

[0008] Figure 1 is a block diagram of one embodiment of a computer system including modifiable revision identification registers.

[0009] Figure 2 is a flow diagram of one embodiment of a method for modifying a value stored in a revision identification register.

[0010] Figure 3 is a flow diagram of an additional embodiment of a method for modifying a value stored in a revision identification register.

Detailed Description

[0011] An embodiment for modifying the contents of a revision identification register includes a revision identification register that is both readable and writable (the contents of the revision identification register are modifiable). A revision identification modification bit is also included. The contents of the revision identification register are only modifiable when the revision identification modification bit is set to indicate that writes to the revision identification register will be accepted. This embodiment is useful in scenarios where the hardware is updated but the operating system device drivers need not be updated for a particular functional unit or particular functional units within a highly-integrated system logic device. In such cases, the pre-operating system boot software (such as the basic input/output system (BIOS) which is executed before the operating system is loaded), which is developed specifically for a particular computer system configuration, can make the decision to place an older revision identification value into the revision identification register before the operating system loads. The operating system assumes that the hardware has not changed. This prevents the unnecessary enumeration, search, and reloading of the device drivers, thereby saving manufacturing time.

[0012] Figure 1 is a block diagram of one embodiment of a computer system including modifiable revision identification registers. The computer system of Figure 1 includes a processor 110. The processor 110 is coupled to a system logic device 120. The system logic device 120 may include a memory controller for communicating with a system memory 130. The system logic device 120 may also include other functional units including a graphics controller.

[0013] The system logic device 120 is also coupled to an input/output hub 140. The input/output hub 140 may include a variety of functional units, including,

but not limited to, functional units that provide communication with a universal serial bus (USB) 165, a PCI bus 155, and a disk drive interface 175. The disk drive interface 175 may communicate with a storage device (not shown) that has stored thereon an operating system. All or part of the operating system may be loaded into the system memory 130 during the computer system boot process.

[0014] The input/output hub 140 is also coupled to a non-volatile memory 180. The non-volatile memory 180 may store a basic input/output system (BIOS) or other pre-operating system software. As used herein, the term "pre-operating system software" is meant to include any software agents that are executed prior to the operating system loading during the system boot process.

[0015] The computer system of Figure 1 includes a number of revision identification registers. The system logic device 120 includes a revision identification register 122 and a revision identification modification bit 124. When the revision identification modification bit 124 is set to the value "1", the revision identification register 122 will accept writes. When the revision modification bit 124 reflects a value of "0", the revision identification register 122 is read-only (will not accept writes).

[0016] The input/output hub 140 includes revision identification registers 141, 143, and 145. The input/output hub 140 also includes revision identification modification bits 142, 144, and 146. The revision identification register 141 and the revision identification modification bit 142 are associated with the functional unit that provides communication with the USB 165. The revision identification register 143 and the revision identification modification bit 144 are associated with the functional unit that provides communication with the PCI bus 155. The revision identification register 145 and the revision identification modification bit 146 are associated with

the functional unit that provides communication with the disk drive interface 175. Many other possible embodiments exist where revision identification registers and revision identification modification bits are associated with any of a wide range of functional units or devices.

[0017] As with the revision identification register 122 and the revision identification modification bit 124, the revision identification registers 141, 143, and 145 are not modifiable when their associated revision identification modification bits 142, 144, and 146 contain the value "0". When the revision identification modification bits 142, 144, and 146 are set to the value "1", the associated revision identification registers 141, 143, and 145 are modifiable. That is, the revision identification registers 141, 143, and 145 will accept writes.

[0018] The pre-operating system software stored in the non-volatile memory 180 may include code that checks the current values stored in the various revision identification registers. This code can determine whether to replace the current revision identification register values with values that reflect previous device steppings.

[0019] For this example embodiment, the revision identification registers 122, 141, 143, and 145 and the revision identification modification bits 124, 142, 144, and 146 may be included as part of the PCI configuration spaces associated with the various associated devices or functional units.

[0020] Although example embodiments described herein discuss a revision identification modification bit for determining whether an associated revision identification register is modifiable, other embodiments are possible using more than one bit.

[0021] Further, although example embodiments described herein include revision identification modification bits where a value of "1" indicates that an associated revision identification register is modifiable and a value of "0" indicates a read-only state for the revision identification register, other embodiments are possible where a value of "0" indicates that an associated revision identification register is modifiable and a value of "1" indicates a read-only state for the revision identification register.

[0022] Figure 2 is a flow diagram of one embodiment of a method for modifying a value stored in a revision identification register. At block 210, a current revision identification register value is read from a revision identification register. A check is made at block 220 to determine whether the current revision identification value indicates a first device stepping. This first device stepping may be the last stepping where any significant changes in functionality occurred. The term "first device stepping" does not necessarily mean the original version of a device or functional unit.

[0023] If the current revision identification value indicates the first device stepping, then no further action is taken. However, if the current revision identification value does not indicate the first device stepping, then at block 230 the current revision identification value is replaced with a value that indicates the first device stepping.

[0024] Figure 3 is a flow diagram of an additional embodiment of a method for modifying a value stored in a revision identification register. At block 310, a pre-operating system software agent is executed. At block 320, a value stored in a revision identification register is read. At block 330, a determination is made as to whether to modify the value stored in the revision identification register. If the

determination is made to not modify the value stored in the revision identification register, then at 350 an operating system is loaded. If the determination is made to modify the value stored in the revision identification register, then the value is modified at block 340 before loading the operating system at block 350. The operations indicated at blocks 320, 330, and 340 are performed under control of the pre-operating system software agent.

[0025] In the foregoing specification the invention has been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are, accordingly, to be regarded in an illustrative rather than in a restrictive sense.

[0026] Reference in the specification to "an embodiment," "one embodiment," "some embodiments," or "other embodiments" means that a particular feature, structure, or characteristic described in connection with the embodiments is included in at least some embodiments, but not necessarily all embodiments, of the invention. The various appearances of "an embodiment," "one embodiment," or "some embodiments" are not necessarily all referring to the same embodiments.